



**John T. Anderson**  
**Engineering Note**

**Date:** Original note written 9/22/98  
**Rev Date:** This new note, replacing A980922A, written 6/12/2000  
**Project:** CFT Front End Axial  
**Doc. No:** A1000612  
**Subject:** Software/Firmware map of 8-MCM Analog Front End board (AFE) and Microcontroller Command List

## Table of Contents

<b>INTRODUCTION.....</b>	<b>1</b>
<b>BOARD ARCHITECTURE – HARDWARE PERSPECTIVE.....</b>	<b>1</b>
<b>BOARD ARCHITECTURE – SOFTWARE PERSPECTIVE.....</b>	<b>1</b>
<b>INTERNAL MEMORY MAP.....</b>	<b>2</b>
OVERVIEW OF MEMORY MAP.....	3
<i>Microcontroller Command Queue Processing Description.....</i>	<i>4</i>
<i>Microcontroller Reboot/Restart.....</i>	<i>4</i>
<i>Microcontroller Command Format.....</i>	<i>5</i>
<i>Microcontroller Status Information.....</i>	<i>6</i>
<i>A word about Microcontroller Speed.....</i>	<i>6</i>
<i>Engineering/ Board Diagnostic Commands Lockout.....</i>	<i>6</i>
ANALOG-TO-DIGITAL (A/D) PARAMETERS, STATUS AND READBACK BLOCK .....	7
<i>Storage Locations Associated with Command 0x01 .....</i>	<i>8</i>
<i>Storage Locations Associated with Command 0x02 and Command 0x10 .....</i>	<i>8</i>
MULTI-CHIP MODULE CONTROL AND STATUS BLOCK (COMMANDS 0x3 THROUGH 0x9).....	10
<i>Storage Locations Associated with Command 0x03 .....</i>	<i>11</i>
<i>Storage Locations Associated with Command 0x04 .....</i>	<i>11</i>
<i>Understanding How to Use Command 0x05.....</i>	<i>11</i>
<i>Storage Locations Associated with Command 0x06 .....</i>	<i>13</i>
<i>Storage Locations Associated with Commands 0x08 and 0x09.....</i>	<i>13</i>
VIRTUAL SVX DATA AND CONTROL BLOCK (COMMANDS 0x0A THROUGH 0x0D).....	14
LVDS CONTROL BLOCK (COMMANDS 0x0E, 0x0F).....	15
STORAGE LOCATIONS ASSOCIATED WITH THE CRYOSTAT CONTROL LOOP (COMMAND 0x13).....	16
MANUAL OVERRIDE OF DAC SETTINGS (COMMAND 0x17).....	17
SELECT 53 MHz CLOCK SKEW (COMMAND 0x18).....	17
PLL ENABLE/DISABLE AND CLOCK TESTING MODES (COMMAND 0x19) .....	17
INTERNAL ENGINEERING DIAGNOSTICS (COMMAND 0x1A) .....	17

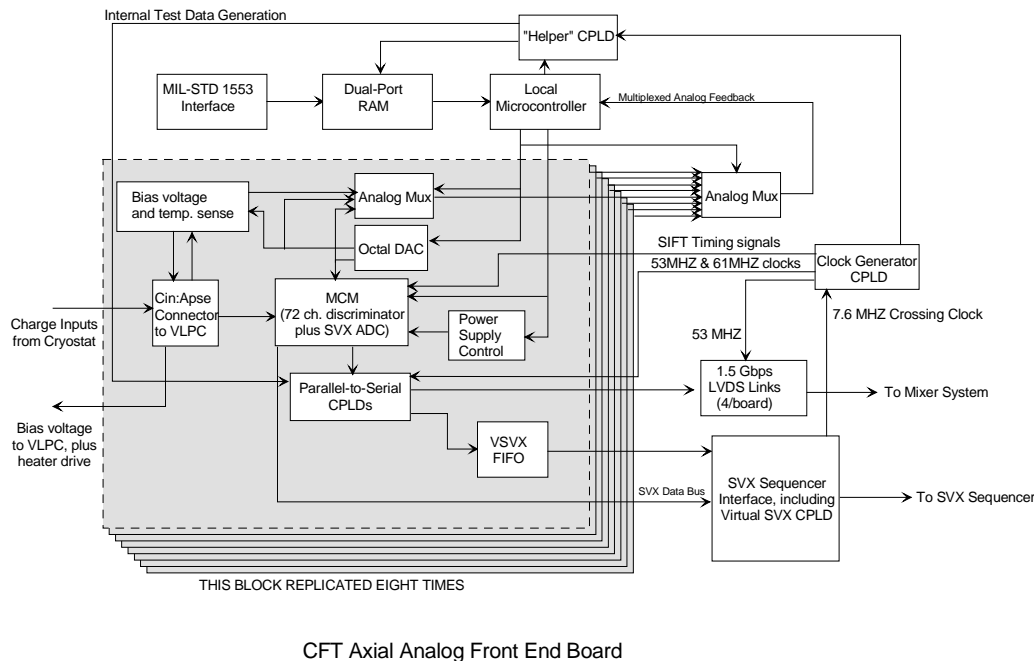
# *Introduction*

The CFT Axial AFE board is connected to the rest of the D0 experiment by a MIL-STD 1553 interface. In this protocol, each CFT board looks like one **Remote Terminal** (RT), with up to 30 registers. Each of the 30 registers should be viewed as a FIFO up to 32 words deep. Whenever a particular register is accessed, it is read from one to 32 times by the host software. Each register is sixteen bits wide.

Jamieson Olsen has written a couple of notes describing the implementation of the 1553 interface; this document attempts to look at the non-1553 side of his logic to show how the various sections of the AFE map out. A full discussion of the microcontroller command structure and a breakdown of how the commands use the memory is within.

## *Board Architecture – Hardware Perspective*

The AFE has a small microcontroller – a PIC14000 – whose basic function is to provide a simplified interface to the various DAC and ADC channels on the board. A secondary function of this microcontroller is to allow each AFE which is installed as the Right-Hand board in a CFT cassette to act as a cassette closed-loop temperature control system. A tertiary function is to provide a control mechanism for sequencing power application to the SIFT and SVX chips on the CFT board. This microcontroller is attached to the main board data bus via a dual-port memory. The dual-port memory allows the 1553 interface to access the microcontroller data in 16-bit chunks, but allows the microcontroller to use the more convenient byte-wide architecture on its side. Further, the dual-port decouples the timing of the two systems. Figure 1 shows a quick sketch of the board architecture, showing the microcontroller and its interface at the top of the picture.



### Figure 1

## *Board Architecture – Software Perspective*

The AFE contains numerous functional blocks, but many of these are only indirectly connected to the software. Thus, the software perspective of the board is very different from the schematic's block-level view. As seen by the software, the CFT Axial card looks like a base address register and a single I/O port.

Register Address	Register Name	Functional Description	Depth on reads	Depth on writes
0	BASE_ADDR	Register which contains base address of window into board memory space, accessed through register 1	One word	One word
1	DATA_PORT	I/O port through which 1553 accesses up to 32 locations of board dual-port RAM	Up to 32	Up to 32

**Table 1**

The 1553 software loads the BASE\_ADDR register with the initial address within the dual-ported RAM that is to be modified, and then writes up to 32 words to the DATA\_PORT register. The 1553 interface keeps a local count of how many words are written or read and drives the internal address bus appropriately. The address written to BASE\_ADDR is the first address in the dual-port RAM which will be read or written by the access to the DATA\_PORT; if the 1553 transaction calls for more than one word to be transferred, the internal address applied to the dual-port RAM is internally incremented with each word transferred through the DATA\_PORT such that a block transfer is effected.

If external software wishes to read or write a block of more than 32 contiguous words of data to the dual-port RAM, the internal address counter should be preserved between accesses to the DATA\_PORT, so simply setting the BASE\_ADDR once and then accessing the DATA\_PORT multiple times should suffice.<sup>1</sup>

## ***Internal Memory Map***

The AFE actually contains 2K X 16 bits of dual-port RAM, which the on-board microcontroller views as 4K X 8. Each table of addresses in the following sections show what the various locations do and gives the address(es) as viewed from both ports of the dual-port RAM. Every 16-bit word in 1553 space maps to two adjacent bytes in microcontroller space. The 'even' byte maps to bits 7..0 of the 16-bit word, with the next-higher 'odd' byte mapping to bits 15..8 of the 16-bit word.

For example: Address 0x57 in 1553 space is the 'command flag' location. From the microcontroller's viewpoint, that 16-bit value is two eight-bit values at addresses 0xAE and 0xAF. Location 0xAE corresponds to bits 7..0 of the 16-bit word, and location 0xAF corresponds to bits 15..8 of the 16-bit word.

The memory map given below is broken in to segments based upon the general set of functions associated with that block of the dual-port RAM space. For each segment, detailed descriptions of each location are given. **Any location listed as reserved or unused must not be written to or read from by external software (e.g., MIL-STD 1553 or event monitoring).**

*Additions or modifications to the memory map must be approved by the keeper of the AFE microcontroller code, currently*

John T. Anderson

Fermilab D-Zero project  
P.O. Box 500 M/S 352  
Batavia, IL 60510  
(630) 840-8885

email: [janderson@fnal.gov](mailto:janderson@fnal.gov)

The latest version of this document is available on the Internet at  
[http://d0server1.fnal.gov/users/janderson/Public\\_Eng\\_Notes/default.html](http://d0server1.fnal.gov/users/janderson/Public_Eng_Notes/default.html)

---

<sup>1</sup> This is the way the logic should work, but in practice all engineering tests have explicitly set the BASE\_ADDR before accessing the DATA\_PORT.

## Overview of Memory Map

The sections to follow will give exact addresses. This table shows the general organization of the dual-port RAM as a quick reference. Unused sections are greyed out, but do *not* assume they are free for use!

Address(es) as programmed from microcontroller	Address(es) as programmed from 1553	Function
0x0000 – 0x00BF	0x0000 – 0x005F	Microcontroller general control and command queue
0x00C0 – 0x00FF	0x0060 – 0x007F	Currently unused.
0x0100 – 0x016F	0x0080 – 0x00B7	A/D calibration, manual A/D conversion data area
0x0170 – 0x01FF	0x00B8 – 0x00FF	Currently unused.
0x0200 – 0x02A1	0x0100 – 0x0150	Multi-chip Module (MCM) control
0x02A2 – 0x02FF	0x0151 – 0x017F	Currently unused.
0x0300 – 0x0306	0x0180 – 0x0182	Virtual SVX (VSVX) control
0x0307 – 0x037F	0x0183 – 0x01BF	Currently unused.
0x0380 – 0x0393	0x01C0 – 0x01C9	LVDS control and status
0x0394 – 0x03FF	0x01CA – 0x01FF	Currently unused.
0x0400 – 0x0469	0x0200 – 0x0234	Cryostat control loop parameters and status
0x046A – 0x04FF	0x0235 – 0x027F	Currently unused.
0x0500 – 0x057F	0x0280 – 0x02BF	Reserved for engineering DAC tests
0x0580 – 0x05FF	0x02C0 – 0x02FF	Currently unused.
0x0600 – 0x0603	0x0300 – 0x0301	Reserved for engineering clock control tests.
0x0604 – 0x06FF	0x0302 – 0x037F	Currently unused.
0x0700 – 0x07FF	0x0380 – 0x03FF	Reserved for engineering memory and register diagnostics.
0x0800 – 0x0BFF	0x0400 – 0x05FF	Buffer for data transfer to VSVX Inbound FIFO
0x0C00 – 0x0FFF	0x0600 – 0x07FF	Buffer for data transfer from VSVX Monitor FIFO

**Table 2**

## Microcontroller Command Processing Memory Block

Upon power up the microcontroller resets and then enters an idle loop. As part of the reset, locations 0x00A0 through 0x00AF (Microcontroller command block) and locations 0x00B0-0x00BF (Microcontroller Status Block) are cleared to zeroes. In the idle loop, the microcontroller polls location 0x00AF, and if it is non-zero, interprets the values found in 0x00A0 through 0x00AE as a command sequence. The commands stored in these locations are processed in order. After the queue is empty the microcontroller re-enters the idle state.

Address(es) as programmed from microcontroller	Address(es) as programmed from 1553	Function
0x0000 – 0x009F	0x0000 – 0x004F	Reserved.
0x00A0 – 0x00AD	0x0050 – 0x0056	Microcontroller command queue. Each word is a separate command. Only the lower byte of each word is examined by microcontroller.
0x00AE	0x0057 (LSByte)	‘Execute Command List’ location. Set to non-zero value to force microcontroller to execute commands previously stored in queue above.
0x00AF	0x0057 (MSByte)	Unused by microcontroller.
0x00B0 – 0x00BD	0x0058 – 0x005E	Reserved for microcontroller internal diagnostics.
0x00BE	0x005F (LSByte)	Microcontroller Command Loopback. Contains value of last command processed by microcontroller.
0x00BF	0x005F (MSByte)	Microcontroller heartbeat. Regularly incremented by microcontroller; if not changing, micro program is stuck and board requires reboot.

Table 3

## Microcontroller Command Queue Processing Description

The microcontroller, when triggered by writing a non-zero value to location 0x0057 (1553 side), begins processing by first reading the data value stored at location 0x0050 (1553 side). The microcontroller executes each command after it is read, then advances to the next location. As a general rule the micro is infinitely fast compared to 1553 speeds, so that the entire list will typically be processed before the 1553 bus can check to see if the first command has been processed.

*The command list must be terminated by a zero.* The microcontroller will continue to process commands in the list until it finds a value of zero, which it interprets as both ‘no-op’ and ‘terminate list’. As a safeguard, the micro clears the trigger location (0x0057 from the 1553 side) to zero as soon as it sees the trigger. An additional safeguard is that each command in the queue is also cleared to zero as it is processed. *Best practice, however, is to always write the entire queue and fill unused commands with zeroes to avoid erroneous operation.*

Although real-time monitoring of the microcontroller is impossible from the 1553 side, because the micro is so much faster, a minor diagnostic is available at location 0x005F (1553 side). The least significant byte of this location will always contain the last command processed by the micro, and the most significant byte will contain a heartbeat that is continuously incremented by the micro. Should the event monitoring system at any time suspect that the microcontroller is ‘hung’, it may interrogate this location.

## Microcontroller Reboot/Restart

The microcontroller reboots upon power up or when the manual reset pushbutton on the board is depressed. Should remote reboot/restart be required for any reason, it must be accomplished by using the Rack Monitor system to shut down the power supply associated with the board in question. A ‘soft reboot’ command is implemented which will restart the microcontroller’s program and reset the various registers and DACs to standard values, assuming that the micro is capable of responding to the command.

## Microcontroller Command Format

Each 16-bit value written to the command queue is interpreted as a single command value for the microcontroller. The lower 5 bits are the actual command, and at present the upper bits are ignored. The commands implemented are as shown in Table 4. The majority of commands read and/or write locations in the dual port RAM. The details of how those sections of RAM are implemented are in the following sections of this document.

Value	Command Function	Notes
0x0	No-op / terminate command list	
0x1	Perform internal calibration of A/D converter.	Updates A/D parameter block in dual port ram.
0x2	Do manual A/D conversion of selected analog input(s). <i>Calling this function disables automatic cryo control loop.</i>	Performs up to 32 A/D conversions dependent upon bitmasks set up prior to command; all A/D conversions stored in fixed block of dual-port ram.
0x3	Set Discriminator Gain	Uses bitmask byte in dual-port RAM to set discriminator gain on MCM-by-MCM basis.
0x4	Set SIFT-to-SVX Charge Transfer Gain	Uses bitmask byte in dual-port RAM to set charge transfer gain on MCM-by-MCM basis.
0x5	Load Discriminator Thresholds	Reads list of 32 bytes (one per SIFT) from dual-port RAM, loads threshold DACs with demanded values.
0x6	Load SVX chip VREF values	Reads list of 32 bytes (one per SIFT) from dual-port RAM, loads VREF DACs with demanded values.
0x7	Fire internal test pulser on board to create test charge into MCMs	Feature may not be available pending results of engineering tests of prototype AFE.
0x8	Turn on power to selected Multi-Chip Modules (MCMs) on board.	Which MCMs is dependent upon MCM select bitmask.
0x9	Turn off power to selected Multi-Chip Modules (MCMs) on board.	Which MCMs is dependent upon MCM select bitmask.
0xA	Transfer block of data from dual-port RAM to Virtual SVX Inbound FIFO	Data is of arbitrary length up to max of 256 bytes; actual length transferred to FIFO controlled by a different dual-port RAM location.
0xB	Transfer current contents of Virtual SVX Monitor FIFO to dual-port RAM	Transfers all data in Monitor FIFO to block at fixed address in dual-port RAM; actual length transferred stored in another location
0xC	Select VSVX Operating Parameters	Loads VSVX control byte from dual-port RAM into VSVX control register 1 per VSVX documentation.
0xD	Set VSVX Event Delay Parameter	Loads delay, in crossings, into VSVX control register 3.
0xE	Upload new LVDS Test Data Seeds	Transfers list of eight seed bytes from dual-port RAM to the eight LVDS Data Mux CPLDs on board; seed used for test data sent during Sync Gap.
0xF	Upload user-definable status into LVDS data stream	Transfers user status bits from dual port RAM into registers of LVDS Data Mux CPLDs, for inclusion in LVDS data.
0x10	Zero out ADC readback values	Clears all ADC readback data to insure that any later reads are fresh
0x11	Zero out board status channels	Clears all non-ADC status information in dual-port ram to insure any later reads are fresh
0x12	Zero out cryo control loop status data	Clears any status left by cryo control loop code
0x13	Enable Cryostat Control Loop	Turns on local temperature control code.
0x14	Disable Cryostat Control Loop and Coast	Turns off local temperature control code; leaves current heater settings untouched and continues to drive heater resistors.
0x15	Disable Cryostat Control Loop and Disable Heaters	Turns off local temperature control code and turns off all heater resistors, allowing VLPCs to cool to minimum temperature.

0x16	Reset all DAC's to zero (bulk reset) and zero out DAC demand section of dual-port RAM <i>Used for engineering tests and on-bench repair only.</i>	Checks for hardware protection key, if key not present, command will not execute.
0x17	Manually override DAC settings on board. <i>Calling this function disables automatic cryo control loop. Used for engineering tests and on-bench repair only.</i>	Reads null-terminated list of DAC channel updates from fixed block in dual-port RAM, overwrites DACs in order based upon this list. Checks for hardware protection key, if key not present, command will not execute.
0x18	Select 53 MHz clock skew <i>Used for engineering tests and on-bench repair only.</i>	Reads byte from dual port RAM and writes it to clock controller control register. Checks for hardware protection key, if key not present, command will not execute.
0x19	Enable/Disable PLLs and/or enter clock generator test mode. <i>Used for engineering tests and on-bench repair only.</i>	Reads byte from dual port RAM and writes it to clock controller control register. Checks for hardware protection key, if key not present, command will not execute.
0x1A	Perform internal engineering diagnostics on dual-port RAM and any other registers. <i>Used for engineering tests and on-bench repair only.</i>	Leaves result if possible in dual-port RAM. Checks for hardware protection key, if key not present, command will not execute.
0x1B – 0x1E	Reserved for future use.	Checks for hardware protection key, if key not present, command will not execute.
0x1F	Request Soft Reboot	Microcontroller will go through reset sequence, to leave board as close to power-up state as possible.

**Table 4**

## Microcontroller Status Information

Memory locations 0x00B0 through 0x00BF are reserved for microcontroller status. Locations 0x00B0 through 0x00BD are used for additional, command-specific information, for use with engineering testing. Location 0x00BE is filled with the last command processed by the microcontroller; should the micro ever hang up, this location may give a clue to what died. Location 0x00BF is a heartbeat location which is simply incremented by the microcontroller every time the code loops through the polling loop that looks for new commands. Since the 1553 interface will be asynchronous to this loop, monitor code should simply look for change in the heartbeat rather than a predictable increment.

## A word about Microcontroller Speed

Simply put, this is not a fast processor. In comparison to 1553, things go quickly, but a typical command will take microseconds or even milliseconds to execute. The processor clock is a 12 MHz clock (~80nsec per instruction), and most commands will require a few hundred to a few thousand instructions. In addition, instruction pipelining is never perfect, so the average instruction time is probably more like 150 nsec each. If a command takes 1000 instructions to perform, that's going to take 150 usec or so.

The really slow commands will be those that require the use of the A/D converter. The A/D inside the microcontroller is a Wilkinson (slope) converter, and so the closer the input signal is to the positive rail, the longer the A/D takes to convert. Each A/D conversion can be expected, on average, to take about 4 milliseconds.

## Engineering/ Board Diagnostic Commands Lockout

Some commands in the table are designed for use at the test bench only and are not appropriate for use when an AFE board is mounted in the experiment. To insure that only reasonable commands are performed on boards in-system, a hardware lockout is built into the microcontroller code. The hardware is a key which is placed on the board when it is at the test bench. The key is physically incompatible with insertion into the cassettes of the Central Fiber Tracker cryostat. The internal firmware of the microcontroller will check for the presence of the key when engineering test commands are seen, and if the key is not there, the micro will simply ignore the command and go to the next one in the queue.

## Analog-to-Digital (A/D) Parameters, Status and Readback Block

The block of dual-port RAM from 0x100 (microcontroller) to 0x13F (microcontroller) is used for all general A/D information. Table 5 breaks it down.

Address(es) as programmed from microcontroller	Address(es) as programmed from 1553	Function
0x0100 – 0x0101	0x0080	Setting of constant current source as 16-bit value (high byte always zero) determined for conversion of 1.3V bandgap source. Filled as a result of executing microcontroller command 0x01, <u>Perform internal calibration of A/D converter.</u>
0x0102 – 0x0103	0x0081	Actual conversion value returned from conversion of bandgap source using current setting in previous word. Filled as a result of executing microcontroller command 0x01, <u>Perform internal calibration of A/D converter.</u>
0x0104 – 0x0105	0x0082	Expected conversion value for conversion of 1.3V bandgap source (if everything was ideal). Filled as a result of executing microcontroller command 0x01, <u>Perform internal calibration of A/D converter.</u>
0x0106 – 0x0107	0x0083	Setting of constant current source as 16-bit value (high byte always zero) determined for conversion of SREFLO bandgap source (nominal 0.13V). Filled as a result of executing microcontroller command 0x01, <u>Perform internal calibration of A/D converter.</u>
0x0108 – 0x0109	0x0084	Actual conversion value returned from conversion of SREFLO bandgap source using current setting in previous word. Filled as a result of executing microcontroller command 0x01, <u>Perform internal calibration of A/D converter.</u>
0x010A – 0x010B	0x0085	Expected conversion value for conversion of SREFLO (nominal 0.13V) bandgap source (if everything was ideal). Filled as a result of executing microcontroller command 0x01, <u>Perform internal calibration of A/D converter.1</u>
0x010c – 0x010D	0x0086	Setting of constant current source as 16-bit value (high byte always zero) determined for conversion of +5V source. Filled as a result of executing microcontroller command 0x01, <u>Perform internal calibration of A/D converter.</u>
0x010E – 0x010F	0x0087	Actual conversion value returned from conversion of +5V supply using current setting in previous word. Filled as a result of executing microcontroller command 0x01, <u>Perform internal calibration of A/D converter.</u>
0x0110 – 0x0111	0x0088	Expected conversion value for conversion of +5V supply bandgap source (if everything was ideal). Filled as a result of executing microcontroller command 0x01, <u>Perform internal calibration of A/D converter.</u>
0x0112 – 0x0113	0x0089	Error flag location. Set to zero (0x0000) if everything works. Each nibble (4-bit group) of this location is set to 0xF to denote that an error occurred. If multiple bit groups are set, multiple errors have occurred.  xxxF indicates an error in the conversion of the 1.3V bandgap reference xxFx indicates an error in the conversion of the SREFLO bandgap reference xFxx indicates that the selected 'best' current source value for 1.3V reference and SREFLO reference conversions was not the same value.  Fxxx indicates overflow error in conversion of +5V supply when converted.  Filled as a result of executing microcontroller command 0x01, <u>Perform internal calibration of A/D converter.</u>
0x0114 – 0x011F	0x008A – 0x008F	Reserved

0x0120	0x0090 (LSByte)	Bitmask for which of the eight VLPC Bias Voltage inputs are to be converted and updated by execution of microcontroller command 0x02, <u>Do manual A/D conversion of selected analog input(s)</u> . A '1' in each bit position tells the micro to convert that channel. The least significant bit corresponds to MCM1, the most significant bit corresponds to MCM8.
0x0121	0x0090 (MSByte)	Unused by microcontroller.
0x0122	0x0091 (LSByte)	Bitmask for which of the eight VLPC Current inputs are to be converted and updated by execution of microcontroller command 0x02, <u>Do manual A/D conversion of selected analog input(s)</u> . A '1' in each bit position tells the micro to convert that channel.
0x0123	0x0091 (MSByte)	Unused by microcontroller.
0x0124	0x0092 (LSByte)	Bitmask for which of the eight Cryostat Temperature inputs are to be converted and updated by execution of microcontroller command 0x02, <u>Do manual A/D conversion of selected analog input(s)</u> . A '1' in each bit position tells the micro to convert that channel.
0x0125	0x0092 (MSByte)	Unused by microcontroller.
0x0126 – 0x0127	0x0093	A/D Conversion of internal temperature sensor, automatically updated any time command 0x01 or 0x02 is executed.
0x0128 – 0x013F	0x0094 – 0x009F	Unused by microcontroller.
0x0140 – 0x016F	0x00A0 – 0x00B7	Storage area for manual conversion of all A/Ds. Filled as a result of executing microcontroller command 0x02, <u>Do manual A/D conversion of selected analog input(s)</u> .

**Table 5**

## Storage Locations Associated with Command 0x01

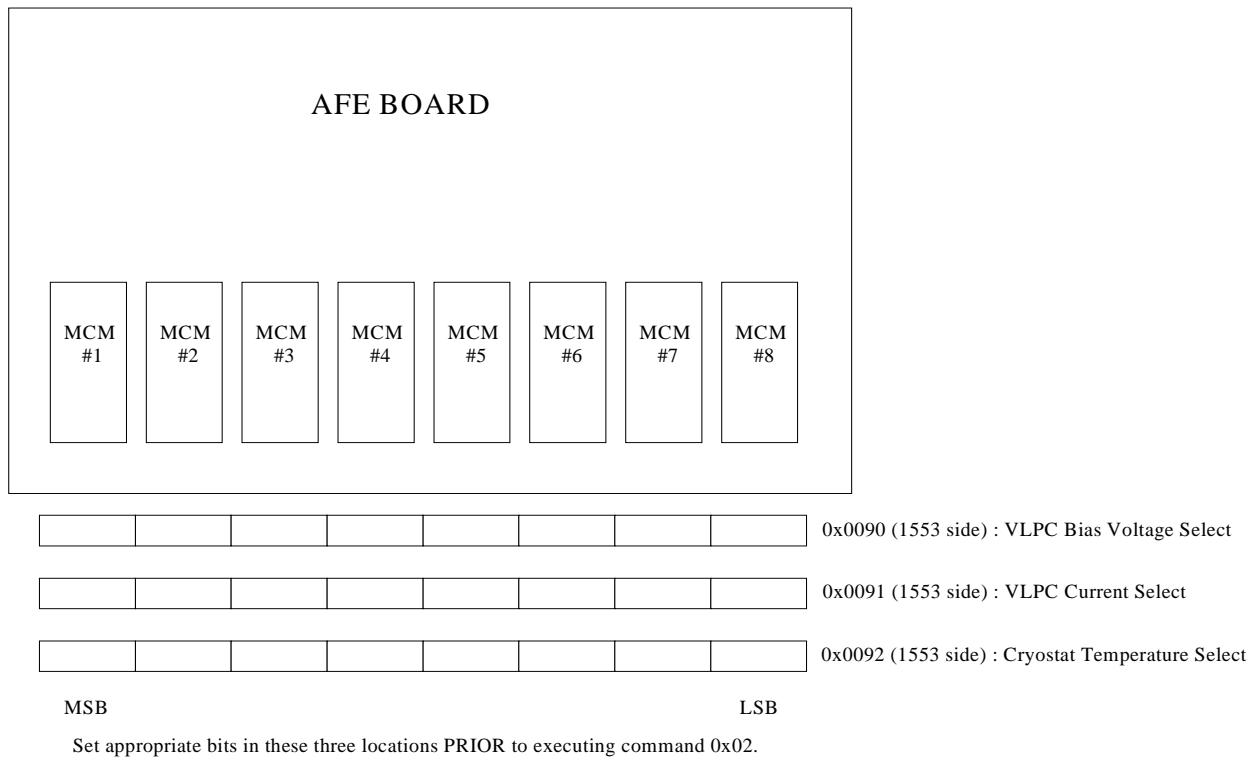
The locations from 0x100 to 0x0113 (microcontroller) are the locations required by Engineering note A1000614, *Internal calibration of A/D converters on AFE, AFE Test, MCM Test and Meltronix boards*, available on the Web at [http://d0server1.fnal.gov/users/janderson/Public\\_Eng\\_Notes/a1000614.pdf](http://d0server1.fnal.gov/users/janderson/Public_Eng_Notes/a1000614.pdf). The values stored in these locations are designed to provide sufficient calibration information such that all other A/D conversions may be appropriately scaled to match one AFE board to another. Location 0x0093 (1553), Board Temperature Sensor, is automatically updated by this command.

## Storage Locations Associated with Command 0x02 and Command 0x10

Microcontroller command 0x02 provides a generic mechanism to read back a selected subset or all of the A/D channels on the board associated with cryostat control. Three locations allow the user to select, using bitmasks, which channels will be converted. The user writes the appropriate bitmasks into locations 0x0090, 0x0091 and 0x0092 (1553 side) and then calls command 0x02 to force the A/D to convert the selected channels.

A block of locations from 0x00A0 to 0x00B7 (1553 side) are reserved to store these conversions, one per bit set in locations 0x0090, 0x0091 and 0x0092 (1553 side). There are eight Flex Cable connections to the AFE board, each one associated with one of the MCMs. The MCMs (and the Flex Cables) are numbered from 1 to 8, with #1 being the left-most cable when the board is being viewed from the component side. *Since AFE boards can be inserted into the cassette as left-handed or right-handed objects, cable #1 may be either the front or the back cable as viewed from the front of the cryostat. Use the least significant bit of the 1553 RT address, which is correlated to backplane position, to determine if the board is right-hand or left-hand and adjust cable count to physical location appropriately.* Figure 2 shows how the bitmasks map to the various physical locations on the AFE board; the view is looking at the board broadside from the component (top) side.

Command 0x02 does the actual conversion work. Command 0x10 may be used prior to executing command 0x02 to clear the readout area to all zeroes, so that new data is more obvious to the casual observer. Location 0x0093 (1553), Board Temperature, is automatically updated whenever command 0x02 is executed.



**Figure 2**

Which locations are updated in the block from 0x00A0 – 0x00B7 (1553 side) are also controlled by these same three bitmask registers. Table 6 shows the relationship between which bits are set and which locations are updated.

Bitmask address	Bit position set	Location updated
0x0090 (1553)	7 (MSB)	0x00A0 (1553)
0x0090 (1553)	6	0x00A1 (1553)
0x0090 (1553)	5	0x00A2 (1553)
0x0090 (1553)	4...0 (LSB)	0x00A3..0x00A7 (1553)
0x0091 (1553)	7 (MSB)	0x00A8 (1553)
0x0091 (1553)	6	0x00A9 (1553)
0x0091 (1553)	5..0 (LSB)	0x00AA..0x00AF (1553)
0x0092 (1553)	7 (MSB)	0x00B0 (1553)
0x0092 (1553)	6..0	0x00B0..0x00B7 (1553)

**Table 6**

The generic algorithm for the execution of command 0x02 is fairly straightforward:

1. Select which channels are to be measured. Reference Figure 2 to help select the ones of interest from physical locations. If desired, use command 0x10 to pre-clear the output data area to all zeroes.
2. Load the appropriate bitmasks into locations 0x0090,0x0091 and 0x0092 (1553 side).
3. Execute command 0x02.
4. Find the results in the dual-port RAM in the block from 0x00A0 – 0x00B7 (1553 side), using Table 4 to match bits set to locations modified.

## Multi-Chip Module Control and Status Block (Commands 0x3 through 0x9)

Commands 0x3 through 0x9 of the microcontroller provide the user with a series of controls all associated with the Multi-Chip modules. A block of dual-port RAM starting at address 0x0200 (microcontroller side, 0x0100 from 1553) provides the necessary interface information for the various commands. Table 7 gives a breakdown of this region.

Address(es) as programmed from microcontroller	Address(es) as programmed from 1553	Function
0x0200	0x0100 (LSByte)	Bitmask byte used by command 0x03 to set discriminator gain (THRESH_SEL) on MCM by MCM basis. MSB corresponds to MCM #1, LSB corresponds to MCM #8, following same convention as used for cryostat controls. This byte selects <i>which</i> MCMs will get affected; the next byte is used to convey whether the selected MCMs have the bit set (1) or cleared (0). Setting a bit here enables the THRESH_SEL for the given MCM to change; a clear bit here leaves the given MCM unchanged regardless of the corresponding bit in the MSByte.
0x0201	0x0100 (MSByte)	Bitmask byte used by command 0x03 along with byte at address 0x200 (micro). For every bit in address 0x200 which is set, the value of the corresponding bit in this byte is written to the selected MCM.
0x0202	0x0101 (LSByte)	Bitmask byte used by command 0x04 to set charge transfer gain (GAIN_SEL) on MCM by MCM basis. MSB corresponds to MCM #1, LSB corresponds to MCM #8, following same convention as used for cryostat controls. This byte selects <i>which</i> MCMs will get affected; the next byte is used to convey whether the selected MCMs have the bit set (1) or cleared (0). Setting a bit here enables the THRESH_SEL for the given MCM to change; a clear bit here leaves the given MCM unchanged regardless of the corresponding bit in the MSByte
0x0203	0x0101 (MSByte)	Bitmask byte used by command 0x04 along with byte at address 0x202 (micro). For every bit in address 0x202 which is set, the value of the corresponding bit in this byte is written to the selected MCM.
0x0204	0x0102 (LSByte)	Bitmask byte used by engineering-test-only commands to set GATEOVR digital control on MCM by MCM basis. MSB corresponds to MCM #1, LSB corresponds to MCM #8, following same convention as used for cryostat controls. <i>This memory location and function has meaning only if the engineering diagnostics key is in place, and has no function in normal use.</i>
0x0205	0x0102 (MSByte)	Unused by microcontroller.
0x0206 – 0x021F	0x0103 – 0x010F	Reserved.
0x0220 – 0x025F	0x0110 – 0x012F	Block of 32 words, one per SIFT, for new discriminator threshold values loaded by command 0x05. Only the least significant 8 bits of each word are valid. <sup>2</sup> The first four words in the list correspond to SIFTs A, B, C and D of MCM #1, the next four to the four SIFTs of MCM #2, etc.
0x0260 – 0x029F	0x0130 – 0x014F	Block of 32 words, one per SIFT, for new SIFT VREF values as loaded by command 0x06. Only the least significant 8 bits of each word are valid. The first four words in the list correspond to SIFTs A, B, C and D of MCM #1, the next four to the four SIFTs of MCM #2, etc.
0x02A0 – 0x02A1	0x0150	MCM select bitmask byte for commands 0x08 and 0x09. MSB corresponds to MCM #1, LSB to MCM #8. Upper byte contains latest board setting.

Table 7

<sup>2</sup> The 8-MCM AFE implements 8 bit DACs for this feature; the 12-MCM version or revisions of the 8-MCM AFE may implement 10 or 12 bit DACs for threshold information, so a full word is reserved for each SIFT. Identical rules apply for the VREF DACs.

## Storage Locations Associated with Command 0x03

The SIFT discriminator implements a dual-slope gain function for the threshold. For whatever analog setting of the threshold is set by command 0x05, one of two different thresholds is selected by the bit programmed through command 0x03. There is one THRESH\_SEL bit per MCM which is common to all four SIFTs in that MCM. Two bytes are used in the dual-port RAM, one which enables change on an MCM-by-MCM basis and one which holds the new values to write if and only if the 'enable change' bit for that MCM has been set. This eliminates the need for any read-modify-write cycles over the 1553 bus. As a convenience to the end user, the on-board microcontroller will automatically save the actual states of the eight THRESH\_SEL bits back into the MSByte of address 0x0100 (1553) after updating all the MCMs.

This may be a little confusing for those who haven't encountered set-reset registers before, so here are the rules:

- The actual status of the THRESH\_SEL bits for all eight MCMs will always be available by reading the MSByte of address 0x0100 (1553) except, of course, when the 1553 software is writing to that location as part of setting up command 0x03.
- To change any THRESH\_SEL bits, the 1553 software must set *two* bits in the word at address 0x0100 (1553) prior to executing command 0x03:
  - Set one bit of bits 7..0 to select the MCM you want to change. Bit 7 == MCM #1, Bit 0 == MCM #8.
  - Set one bit of bits 15..8 (the bit eight positions higher than the select bit you just set) to either 0 or 1 depending on the new state desired for the THRESH\_SEL of the selected MCM.
  - Execute command 0x03 to update all the bits.
- After executing command 0x03, the micro will store the actual settings of all eight MCM THRESH\_SEL bits back in the upper byte (bits 15..8) of address 0x0100 (1553).

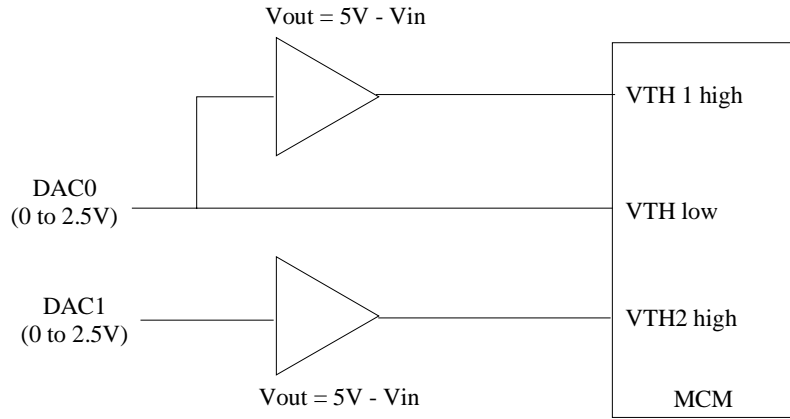
## Storage Locations Associated with Command 0x04

In similar vein to the threshold select manipulated by command 0x03, the charge transfer circuitry in the MCM which connects the four SIFTs to the SVX chip allows for a one-bit selection that uses a smaller or larger capacitor to yield a dual-slope function. The GAIN\_SEL bit is common to all four SIFTs in a single MCM. The exact same set-reset register rules apply for command 0x04 as for command 0x03.

## Understanding How to Use Command 0x05

Command 0x05 reads a table of DAC values from the dual-port RAM and writes them to the DAC channels that control the threshold voltages for each MCM. Each MCM contains four SIFT chips, each of which has its own threshold voltage. However, the way the MCM is wirebonded, and the circuitry of SIFT, require the hardware to provide *two* voltages to the chip, and the threshold is the *difference* between the two voltages. There's also an offset in the voltage – both the low and high parts have to be offset above zero volts – so the interface is not trivial. Limitations in the number of DAC channels which can be put on the board force a compromise.

The compromise which is implemented is shown in Figure 3. One DAC voltage per SIFT is implemented on the board, but adjacent pairs are coupled. The first DAC voltage generates both the low-side of the threshold for two SIFTs *and* the high-side for the first SIFT of the pair. The second DAC voltage generates just the high-side of the second SIFT. Thus, the first DAC voltage completely defines the threshold for the first SIFT but *both* DAC values are part of the equation for the threshold of the second SIFT.



**Figure 3**

Each DAC provides a control voltage from 0 to 2.5 Volts. With the eight bit DACs currently implemented, that's approximately 10 mV per count. The first DAC in each pair provides direct control over the threshold of one SIFT in the MCM. As the DAC voltage is programmed from 2.5V down towards 0V, the inverting op-amp buffer will generate a mirror voltage from 2.5V up to 5V, resulting in the threshold voltage being applied to the SIFT running from 0 to 5V (the difference between the DAC and and op-amp). This results in a net programming resolution in SIFT threshold of about 19.6 mV per DAC count. In terms of SIFT resolution, that's around 1.5 fCoul per DAC count.

$$SIFT1\_threshold = \left( \frac{255 - DAC1\_setting}{255} \right) * 5V$$

For example, if DAC1 is set to code 200, resulting in a SIFT1 threshold of 1.078 volts. *Note that the slope is negative.* As the DAC code written *increases*, the SIFT threshold *decreases*. The maximum SIFT threshold occurs at DAC value zero, reducing to a SIFT threshold of zero at a DAC value of 255 (because both the DAC and the opamp are at 2.5V). Here's a short table of values for an example:

DAC setting	DAC out	opamp out	SIFT1 thresh
0	0.00V	5.00V	5.00V
10	0.10V	4.90V	4.80V
30	0.29V	4.71V	4.41V
90	0.88V	4.12V	3.24V
120	1.18V	3.82V	2.65V
170	1.67V	3.33V	1.67V
255	2.50V	2.50V	0.00V

**Table 8**

The second DAC, after it goes through its op-amp, also generates a voltage from 2.5V up to 5V, but this one is referenced to the output of the first DAC, not to itself. Thus, the threshold voltage applied to the second SIFT follows a more complex equation:

$$SIFT2\_threshold = \left[ 5V * \left( \frac{255 - DAC2\_setting}{255} \right) \right] - \left[ 2.5V * \left( \frac{DAC1\_setting}{255} \right) \right]$$

This setup allows the SIFT2 threshold to be adjusted with respect to the SIFT1 threshold, but the *intent* is that the difference between the SIFT1 threshold and the SIFT2 threshold always be small. A short table or two illustrating what happens will point this out. In this first example, DAC2 is swept over a wide range while DAC1 is kept constant. Adjustment occurs but setting SIFT2's threshold very small while SIFT1's threshold is large is not possible.

DAC1 setting	DAC1 out	DAC2 setting	opamp out	SIFT1 thresh	SIFT2 thresh
20	0.20	0	5.00	4.61	4.80
20	0.20	10	4.90	4.61	4.71
20	0.20	30	4.71	4.61	4.51
20	0.20	90	4.12	4.61	3.92
20	0.20	170	3.33	4.61	3.14
20	0.20	255	2.50	4.61	2.30

**Table 9**

If the circuit is used as intended, as shown here, setting DAC2 close to DAC1 allows a slight variance in threshold between the two SIFT chips. A side benefit is that, since the voltage resolution of the two parts of SIFT2's threshold are different, it's actually possible to achieve 10 mVolt offsets between the two SIFTs, better than the threshold resolution of SIFT1 alone.

DAC1 setting	DAC1 out	DAC2 setting	opamp out	SIFT1 thresh	SIFT2 thresh	Threshold difference
250	2.451	248	2.569	0.098	0.118	-0.020
250	2.451	249	2.559	0.098	0.108	-0.010
250	2.451	250	2.549	0.098	0.098	0.000
250	2.451	251	2.539	0.098	0.088	0.010
250	2.451	252	2.529	0.098	0.078	0.020
250	2.451	253	2.520	0.098	0.069	0.029

**Table 10**

The table of values stored in dual-port RAM locations 0x0110 – 0x012F (1553 side) are mapped to the MCMs and the DAC channels linearly. The first four locations map to MCM #1, the next four to MCM #2, etc., and within each block of four, the order is DAC1, DAC2, DAC3, DAC4, which correspond to SIFT1, SIFT2 (offset), SIFT3 and SIFT4 (offset) per the discussion above. Thus, dual-port RAM location 0x0119 corresponds to MCM #2, SIFT2 (offset).

## Storage Locations Associated with Command 0x06

This command reads the block of data from the dual port RAM and updates the four DACs used to set the VREF voltages which provide an offset between each SIFT chip in the MCM and the single SVX chip there. These DACs are much simpler than the threshold circuit of command 0x05, in that each DAC independently runs from 0 to 2.5V as the code written increases from 0 to 255. The data order is the same as for command 0x05, save that the data block is located at dual-port RAM addresses 0x0130 – 0x014F (1553). The first four locations of the block map to MCM #1, the next four to MCM #2, etc., and within each block of four, the order is DAC1, DAC2, DAC3, DAC4, which correspond to SIFT1\_VREF, SIFT2\_VREF, SIFT3\_VREF and SIFT4\_VREF.

## Storage Locations Associated with Commands 0x08 and 0x09

Commands 0x08 and 0x09 turn the power to the MCMs on and off, respectively. As it is likely that event monitor software will want to manipulate the MCMs independently, the word at location 0x0150 (1553) contains two pieces of information. The lower byte is written to by the 1553 interface to select MCMs which are to be affected (1 == change MCM power status, 0 == leave power status unchanged); the command itself selects whether the list of MCMs are turned on or off. The upper byte is filled by the microcontroller once all MCMs are updated and contains a bitmask of which MCMs are on, and which are off. This allows the event monitor to check status at any time.

**The microcontroller reserves the right to turn any MCM off by itself without receipt of command if an unsafe condition exists. The status byte of location 0x0150 (1553) will be updated should this occur.**

## Virtual SVX Data and Control Block (Commands 0x0A through 0x0D)

The Virtual SVX circuitry of the AFE is dissected in engineering note A1000420, *SVX and Virtual SVX operation in the Analog Front End Board*, located on the Web at

[http://d0server1.fnal.gov/users/janderson/Public\\_Eng\\_Notes/a1000420.pdf](http://d0server1.fnal.gov/users/janderson/Public_Eng_Notes/a1000420.pdf)

The Virtual SVX provides a set of functions connecting the local microcontroller to the readout and control of the SVX chips located in the multi-chip modules. Two FIFOs connect these disparate parts of the board, the **Inbound FIFO** and the **Monitor FIFO**. The Inbound FIFO provides a method for the microcontroller to put data into the SVX readout stream for cable testing, bit error rate checks or to pad the readout with board status information. The Monitor FIFO allows microcontroller access to data that normally flows only through the SVX readout, allowing diagnostics when the SVX readout fails. Areas in the dual-port RAM are reserved for data to/from these FIFOs.

The Virtual SVX also implements three control registers, one of which is reserved for engineering diagnostics. A couple of bytes in the dual-port RAM shadow the user registers allowing the 1553 interface to control the setup and operation of the VSVX. Table 11 shows the RAM locations reserved for all VSVX-related functions.

Address(es) as programmed from microcontroller	Address(es) as programmed from 1553	Function
0x0300	0x0180 (LSByte)	Byte for storage of data value to write to VSVX Control Register 1 when command 0x0C is executed. Different bits have different meanings; refer to engineering note A1000420 for details.
0x0301	0x0180 (MSByte)	Unused by microcontroller.
0x0302	0x0181 (LSByte)	Byte for storage of data value to write to VSVX Control Register 3 when command 0x0D is executed. This data value should match the delay, in crossings, which is programmed into the SVX chips themselves during initialization.
0x0303	0x0181 (MSByte)	Unused by microcontroller.
0x0304 – 0x0305	0x0182	Count of bytes of data to transfer from dual-port RAM to Inbound FIFO when command 0x0A is executed. Data to transfer comes from block later in memory; see rest of this table for addresses. Maximum count is 512 bytes, limited by depth of actual FIFO chip. <sup>3</sup>
0x0306 – 0x0307	0x0183	Count of bytes of data to transfer from Monitor FIFO to dual-port RAM when command 0x0B is executed. Data is transferred to a block later in memory; see rest of this table for addresses. Maximum count is 512 bytes, limited by depth of actual FIFO chip.
0x0800 – 0x09FF	0x0400 – 0x04FF	512 byte block for data to store into Inbound FIFO when command 0x0A is executed.
0x0A00 – 0x0BFF	0x0500 – 0x05FF	Reserved extension to 1K bytes in case larger FIFOs are used.
0x0C00 – 0x0DFF	0x0600 – 0x06FF	512 byte block for storage of data read from Monitor FIFO when command 0x0B is executed.
0x0E00 – 0x0FFF	0x0700 – 0x07FF	Reserved extension to 1K bytes in case larger FIFOs are used.

**Table 11**

<sup>3</sup> Variations of the FIFO chip with identical pinout but larger storage capacity are available and may be used in later revisions of the AFE board. To allow for this, large sections of dual-port RAM adjacent to the storage areas have been reserved.

## LVDS Control Block (Commands 0x0E, 0x0F)

The AFE board contains eight LVDS Data Mux components, each of which is implemented in a programmable logic device. In normal operation these devices capture the data from the MCMs and serialize the bits for transmission over LVDS cables to the trigger system. During beam gaps the AFE board fills the links with test pattern data to allow for background bit error or cable error checks. The test pattern data is psuedo-random, generated from a seed value which is loaded by the microcontroller in response to command 0x0E. Eight bytes of dual-port RAM are allocated for this function, one per PLD. The first byte corresponds to the PLD which works with MCM #1 (using the naming convention first displayed as Figure 2); the last byte to the PLD tied to MCM #8.

A mismatch in the number of bits that *are* transmitted by the LVDS links of the AFE and the number of bits *required* allows for the possibility of including some excess status information in each board, perhaps to identify which board sourced the data, or perhaps to mark some state of the experiment. A total of 512 bits are needed to send all the MCM data each beam crossing, but during that time the LVDS links can send 560 bits, leaving 48 as ‘extras’. A number of these are consumed by timing and AFE state tag bits, but some should still be left over for user status. The exact number of user bits is still to be determined based upon test results obtained from the AFE and DFE prototypes. A block of 32 bits (four bytes) are set up in the dual-port RAM for this purpose, and the exact number will be set in the next revision of this document.

Address(es) as programmed from microcontroller	Address(es) as programmed from 1553	Function
0x0380 – 0x038F	0x01C0 – 0x01C7	Block of eight words, the LSByte of each are the ‘seed’ values for the psuedo-random data generators of the eight LVDS Data Mux CPLDs. Address 0x01C0 corresponds to MCM#1, 0x01C7 to MCM #8. Only the LSByte is used; the MSByte of each word is ignored. These values are copied to the Data Mux parts when command 0x0E is executed.
0x0390 – 0x0393	0x01C8 – 0x01C9	Block of four bytes (two words), that contain 32 bits which may be added to AFE status information when command 0x0F is executed. <i>The actual function of command 0x0F is as yet indeterminate. Further clarification will follow after the prototype AFE boards have been tested.</i>

Table 12

## Storage Locations Associated with the Cryostat Control Loop (Command 0x13)

Command 0x13 enables the Cryostat Control Loop firmware, which uses the on-board microcontroller to continuously monitor and update the temperature of the cryostat upon which the AFE board sits. Stefan Grunendahl is writing this piece of the firmware, and the memory block described in Table13 is based upon his preliminary notes. For details of what different values do, the reader is referred to Stefan's documentation when it becomes available.

Address(es) as programmed from microcontroller	Address(es) as programmed from 1553	Function
0x0400 – 0x0409	0x0200 – 0x0204	Various control and timing parameters affecting general operation of cryostat control loop software. Location 0x0409 (micro) is unused. Further details TBD, see Stefan for documentation.
0x040A – 0x0415	0x0205 – 0x020A	PID parameters associated with Flex Cable #1: 0x0205: set point 0x0206 (LSByte): proportional gain parameter 0x0206 (MSByte): integral gain parameter 0x0207 (LSByte): derivative gain parameter 0x0207 (MSbyte): output variable 0x0208 : process variable (current) 0x0209: integral term value 0x020A: derivative term value
0x0416 – 0x0421	0x020B – 0x0210	PID parameters associated with Flex Cable #2 (see breakdown above)
0x0422 – 0x042D	0x0211 – 0x0216	PID parameters associated with Flex Cable #3 (see breakdown above)
0x042E – 0x0439	0x0217 – 0x021C	PID parameters associated with Flex Cable #4 (see breakdown above)
0x043A – 0x0445	0x021D – 0x0222	PID parameters associated with Flex Cable #5 (see breakdown above)
0x0446 – 0x0451	0x0223 – 0x0228	PID parameters associated with Flex Cable #6 (see breakdown above)
0x0452 – 0x045D	0x0229 – 0x022E	PID parameters associated with Flex Cable #7 (see breakdown above)
0x045E – 0x0469	0x022F – 0x0234	PID parameters associated with Flex Cable #8 (see breakdown above)

**Table 13**

### **Manual Override of DAC Settings (Command 0x17)**

This command reserves the block of RAM from address 0x0500 to 0x05FF (micro) for a list of DAC values and commands which are processed as a sequential list of DAC updates. Each word is interpreted as a DAC address to update and the new value to set that DAC to. Up to 128 updates may be accomplished through this command. *This command is only available for engineering tests and will not execute unless the hardware key is in place.*

### **Select 53 MHz Clock Skew (Command 0x18)**

The byte at address 0x0600 (micro) is reserved for data associated with this command.

### **PLL Enable/Disable and Clock Testing Modes (Command 0x19)**

The byte at address 0x0602 (micro) is reserved for data associated with this command.

### **Internal Engineering Diagnostics (Command 0x1A)**

The data block from address 0x0700 through 0x07FF (micro) is reserved for this command.

Address(es) as programmed from microcontroller	Address(es) as programmed from 1553	Function
0x0500 – 0x05FF	0x0280 – 0x02FF	Block reserved for manual DAC setting engineering routines (Command 0x17)
0x0600	0x0300 (LSByte)	Clock skew control byte, for use with command 0x18. <i>Engineering use only, requires hardware key.</i>
0x0601	0x0300 (MSByte)	Unused by microcontroller.
0x0602	0x0301 (LSByte)	PLL enable/disable, Clock testing control byte for use with command 0x19. <i>Engineering use only, requires hardware key.</i>
0x0603	0x0301 (MSByte)	Unused by microcontroller.
0x0604 – 0x06FF	0x0302 – 0x037F	Available for use.
0x0700 – 0x07FF	0x0380 – 0x03FF	Reserved for internal board diagnostic routines (Command 0x1A)

**Table 14**